CSc 174 Database Management Systems

12. Introduction to Transaction Processing Concepts and Theory

Ying Jin
Computer Science Department
California state University, Sacramento

Transaction

- Operations to access a database
  - read –retrieval
  - write - insert or update
  - Delete
- **Transaction**
  - A logical unit of database processing that includes one or more access operations

Simple Model of a Database

For purposes of discussing transactions

- A database
  - collection of named data items
- Granularity of data
  - a field, a record , or a whole disk block
- Basic operations are read and write
  - read_item(X)
  - write_item(X)

Read

- **read_item(X):**

  Reads a database item named X into a program variable. To simplify our notation

  we assume that *the program variable (in main memory) is also named X.*

- **read_item(X) command includes the following steps:**

1. Find the address of the disk block that contains item X.
2. Copy that disk block into a buffer in main memory (if that disk block is not already in some main memory buffer).
3. Copy item X from the buffer to the program variable named X.

Write

- write_item(X)

  Writes the value of program variable X into the database item named X.

Write (Cont.)

- **write_item(X) command includes the following steps:**

1. Find the address of the disk block that contains item X.
2. Copy that disk block into a buffer in main memory (if that disk block is not already in some main memory buffer).
3. Copy item X from the program variable named X into its correct location in the buffer.
4. Store the updated block from the buffer back to disk (either immediately or later).

Examples

(a)

T_1

```
read_item (X);
X:=X-N;
write_item (X);
read_item (Y);
Y:=Y+N;
write_item (Y);
```

(b)

T_2

```
read_item (X);
X:=X+M;
write_item (X);
```

More about a transaction

- A **transaction** is an atomic unit of work that is either completed in its entirety or not done at all.
- Balance Transfer Example

Properties of Transactions

ACID properties:

- **Atomicity**: A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.
- **Consistency preservation**: A correct execution of the transaction must take the database from one consistent state to another.
  - A consistent state of the database satisfies the constraints specified in the schema as well as any other constraints that should hold on the database.

Properties of Transactions (Cont.)

ACID properties (cont.):

- **Isolation**: A transaction should appear as if it is being executed in isolation from other transactions. That is, the execution of a transaction should not be interfered with by any other transactions executing concurrently.
- **Durability or permanency**: Once a transaction changes the database and the changes are committed, these changes must never be lost because of subsequent failure.

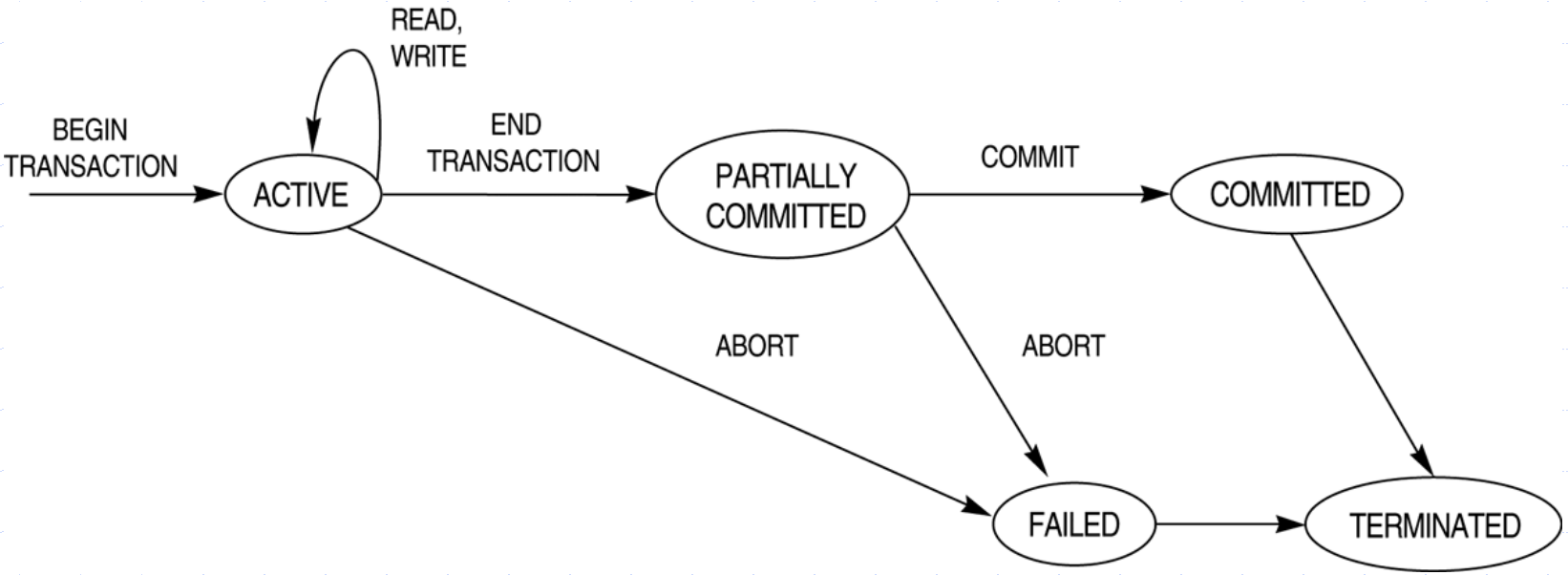
Information of a Transaction

- Important information of a transaction
  - **begin_transaction:** This marks the beginning of transaction execution.
  - **read or write:** These specify read or write operations on the database items that are executed as part of a transaction.
  - **end_transaction:** All the read and write transaction operations have ended

Information of a Transaction (Cont.)

- **commit_transaction:** This signals a *successful end* of the transaction so that any changes (updates) executed by the transaction can be safely load to the database
- **Rollback (or abort):** This signals that the transaction has *ended unsuccessfully*, so that any changes or effects that the transaction may have applied to the database must be *undone*.

States for transaction execution



Failure

- Can a transaction fail?
- Failure modes

Failure Modes

- Erroneous Data Entry
  - E.g. Mistypes one digit of a phone number.
- Media Failures
  - E.g. head crashes
- Catastrophic Failure
  - Media completely destroyed.
  - E.g. fire

Failure Modes (Cont.)

- System Failures
  - E.g. power loss, software errors
  - Data in memory lost
  - State of transaction lost
- Failure -> recovery needed!

The system Log

- Keeps track of all transaction operations that affect the values of database items.
- Use log to recovery from transaction failures.
- The log is kept on disk, so it is not affected by any type of failure except for disk or catastrophic failure.
- The log is periodically backed up to archival storage (tape) to guard against such catastrophic failures.

Types of Log record

1. [start_transaction,T]: Records that transaction T has started execution.
2. [write_item,T,X,old_value,new_value]: Records that transaction T has changed the value of database item X from old_value to new_value.
3. [read_item,T,X]: Records that transaction T has read the value of database item X.
4. [commit,T]: Records that transaction T has completed successfully, and affirms that its effect *can be* committed (recorded permanently) to the database.
5. [abort,T]: Records that transaction T has been aborted.

Force writing a log

- *Before* a transaction reaches its commit point, any portion of the log that has not been written to the disk yet must now be written to the disk.

- How to use log for recovery?

  - Lecture 12.

These slides are based on the textbook:

R. Elmasri and S. Navathe, *Fundamentals of Database System*, 7th Edition, Addison-Wesley.